UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/864,439	05/24/2001	John R. Applin	10004626-1	2655

22879 7590 02/28/2005

HEWLETT PACKARD COMPANY
P O BOX 272400, 3404 E. HARMONY ROAD
INTELLECTUAL PROPERTY ADMINISTRATION
FORT COLLINS, CO 80527-2400

EXAMINER	
KANG, INSUN	
ART UNIT	**PAPER NUMBER**
2124	

DATE MAILED: 02/28/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/864,439	APPLIN, JOHN R.	
	Examiner	Art Unit	
	Insun Kang	2124	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *8/3/2004*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1,6-8,10,11,17 and 21-30* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1, 6-8, 10, 11, 17, and 21-30* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *03 August 2004* is/are: a)☐ accepted or b)☒ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is in response to the amendment filed 8/3/2004.

2. As per applicant's request, claims 2-5, 9, 12-16, and 18-20 have been cancelled, claims 1, 8, 10, 11, and 17 have been amended and claims 21-30 have been added. Claims 1, 6-8, 10, 11, 17, and 21-30 are pending in the application.

Drawings

3. Regarding the syntax of ***C operator* () const {return *operator -> ();}*** in Fig 2, applicant's argument is persuasive because the function returns the dereferenced variable.

4. The drawings filed 8/3/2004 are objected to under 37 CFR 1.83(a). The drawings must show every feature of the invention specified in the claims. Therefore, the amended claims 1, 11, and 17 must be shown or the feature(s) canceled from the claim(s). No new matter should be entered.

Corrected drawing sheets in compliance with 37 CFR 1.121(d) are required in reply to the Office action to avoid abandonment of the application. Any amended replacement drawing sheet should include all of the figures appearing on the immediate prior version of the sheet, even if only one figure is being amended. The figure or figure number of an amended drawing should not be labeled as "amended." If a drawing figure is to be canceled, the appropriate figure must be removed from the replacement sheet, and where necessary, the remaining figures must be renumbered and appropriate changes made to the brief description of the several views of the drawings for consistency. Additional replacement sheets may be necessary to show the renumbering

of the remaining figures. Each drawing sheet submitted after the filing date of an application must be labeled in the top margin as either "Replacement Sheet" or "New Sheet" pursuant to 37 CFR 1.121(d). If the changes are not accepted by the examiner, the applicant will be notified and informed of any required corrective action in the next Office action. The objection to the drawings will not be held in abeyance.

Claim Rejections - 35 USC § 101

5. The rejection to claim 11 has been withdrawn due to the amendment to the claim.

Claim Objections

6. Claim 25 is objected to because of the following informalities: It appears that claim 25 needs to be corrected to claim 30. Appropriate correction is required.

Claim Rejections - 35 USC § 112

7. The rejection to claims 1, 6-8, 10, and 17 has been withdrawn due to the amendment to the claims.

Claim Rejections - 35 USC § 102

8. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

9. Claims 21-23, 25-28, and 30 are rejected under 35 U.S.C. 102(b) as being anticipated by Pike et al. ("Checkmate: Cornering C++ Dynamic Memory Errors With Checked Pointers," 3/2000) hereinafter referred to as "Pike."

Per claim 21:

Pike anticipates:

-storing and manipulating pointers in a programming language, wherein the programming language supports a native pointer type and standard pointer operations upon a native pointer object of said native pointer type ("checked pointers, a simple wrapper for C++ pointers that prevents pointer arithmetic and other common sources of pointer errors, and detects all ...errors...The syntax of checked pointers is highly faithful to raw C++ pointers, but provides run-time error detection and debugging information," abstract; see also Class Pointer definition in Pointer_2.h of the technical addendum of this article)

-defining a safe pointer type supporting said standard pointer operations (see the class definition of "Pointer" type supporting pointer operations such as * (dereferencing operator) and -> (indirection operator) in pg 355 left column);

-performing automatic pointer checking in association with safe pointer type by checking for improper pointer alignment ("checked pointers...prevents pointer arithmetic and other common sources of pointer errors, and detects all dereferencing and deallocation errors, including memory leaks, abstract) as claimed.

Per claim 22:

The rejection of claim 21 is incorporated, and further, Pike discloses performing error processing ("detects and reports common bugs such as illegal dereferences," pg 352 left column, Introduction section) as claimed.

Per claim 23:

The rejection of claim 22 is incorporated, and further, Pike discloses that said error processing includes at least one of generating an error message and terminating program execution; generating a warning message without terminating program execution; and invoking a user defined error processing routine ("terminate execution immediately and output a meaningful error message," pg 354 left column; See also operator ->() and operator* () functions in Pointer_2.h of the technical addendum of this article) as claimed.

Per claim 25:

The rejection of claim 21 is incorporated, and further, Pike discloses reading a global variable of said safe pointer type (the class Pointer is template class, see the class Pointer definition in pg 355 left column) said global variable operable to encapsulate a misaligned pointer as an improper native pointer object; performing said standard pointer operation upon said object (the class Pointer is template class, see the class Pointer definition in pg 355 left column) and performing error processing in response to

said improper native pointer object (the class Pointer is template class, see the class Pointer definition in pg 355 left column; "keep track of where each pointer is pointing, and which memory locations are currently allocated. We check at run-time that a pointer points t a legal address before deleting or dereferencing it. If the check fails, execution halts and the error is reported immediately," pg 355, left column, paragraph 4) as claimed.

Per claims 26-28, and 30, they are the program product versions of claims 21-23, and 25, respectively, and are rejected for the same reasons set forth in connection with the rejection of claims 21-23, and 25 above.

Claim Rejections - 35 USC § 103

10. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

11. Claims 1, 6-8, 10, 11, and 17 are rejected under 35 U.S.C. 103(a) as being unpatentable over Pike et al. ("Checkmate: Cornering C++ Dynamic Memory Errors With Checked Pointers," 3/2000) hereinafter referred to as "Pike" in view of Morris ("Objects First," 1998).

Per claim 1:

Pike anticipates:

-storing and manipulating pointers in a programming language, wherein the programming language supports a native pointer type and standard pointer operations upon a native pointer object of said native pointer type ("checked pointers, a simple wrapper for C++ pointers that prevents pointer arithmetic and other common sources of pointer errors, and detects all ...errors...The syntax of checked pointers is highly faithful to raw C++ pointers, but provides run-time error detection and debugging information," abstract; see also Class Pointer definition in Pointer_2.h of the technical addendum of this article)

-defining a safe pointer type supporting said standard pointer operations (see the class definition of "Pointer" type supporting pointer operations such as * (dereferencing operator) and -> (indirection operator) in pg 355 left column);

-performing automatic pointer checking in association with said safe pointer type by checking for a null pointer ("checked pointers...that detects all dereferencing and deallocation errors, including memory leaks," abstract; ("We detect dereferences of null and dangling pointers by recognizing that requested memory addresses are not in this set," pg 354 left column; see also the implementation of "checked pointer," in Pointer_2.h file of the technical addendum of this article)

Pike does not explicitly teach performing error processing by performing **at least one of**: generating a warning message without terminating program execution; and invoking a user defined error processing routine. However, Morris teaches such a

programming technique was a well-known basic syntax feature in the art of software development, at the time applicant's invention was made, to provide a graceful action to a user upon detecting an error (Morris, i.e. page 1) such as those in Morris. Morris teaches the C++ exit() (in assert function) and return functions. Exit function is used to immediately end the program execution (i.e. page 2) while return is used to end only the function execution (i.e. page 1). In Pike, the program is terminated after generating an error message such as that in the fig 2 of the present invention. In order to prevent such a program termination, the only change to make is to delete the exit function. A return function (no action or any other user defined actions) can be replaced accordingly. Therefore, it would have been obvious for one having ordinary skill in the art of computer software development to modify Pike's disclosed system to incorporate the teachings of Morris. The modification would be obvious because one having ordinary skill in the art would be motivated to provide an error message to a user without terminating the program execution so that the user can continue to perform an appropriate action upon the error suggested by Morris (i.e. page 1).

Per claim 7:

The rejection of claim 1 is incorporated, and further, Pike discloses reading a global variable of said safe pointer type; and performing said standard pointer operation upon said object (the class Pointer is template class, see the class Pointer definition in pg 355 left column) as claimed.

Per claim 10:

The rejection of claim 1 is incorporated, and further, Pike discloses reading a global variable encapsulating an improper native pointer object comprising a null pointer; and performing error processing in response to said improper native pointer object (the class Pointer is template class, see the class Pointer definition in pg 355 left column; "keep track of where each pointer is pointing, and which memory locations are currently allocated. We check at run-time that a pointer points t a legal address before deleting or dereferencing it. If the check fails, execution halts and the error is reported immediately," pg 355, left column, paragraph 4) as claimed.

Per claim 11, it is the safe pointer class version of claim 1, respectively, and is rejected for the same reasons set forth in connection with the rejection of claim 1 above.

Per claim 17, it is the product version of claim 1, respectively, and is rejected for the same reasons set forth in connection with the rejection of claim 1 above.

Per claim 6:

The rejection of claim 1 is incorporated, and further Pike and Morris do not explicitly teach calling a function which returns as its value an object of said safe pointer type. Official Notice is taken that calling a function with returning statement, within another function is well known in the art of software development, at the time applicant's

invention was made. This programming technique is used to make the program easier to understand, modify, avoid redundant coding, reuse, test and debug by breaking a program's task into subtasks, solving these subtasks by sub-algorithms and then calling the subtasks. It would have been obvious for one skilled in the art of computer software development to modify Pike's disclosed T& operator* () function which returns as its value an object of the Pointer class to reuse the T* operator->() by simply calling this function and dereferencing the returned pointer to an object, as this function has a same task of "detect[ing] and report[ing] common bugs such as illegal dereferences (pg 352, left column)" with that in T* operator->(). The modification would be obvious because one skilled in the art would be motivated to reuse the same code portions without repeating the same in another function.

Per claim 8:

The rejection of claim 1 is incorporated, and further, Pike and Morris do not explicitly teach calling a function which returns as its value an object of said safe pointer type. However, see the rejection of the claim 6 above.

Pike further discloses: object encapsulating an improper native pointer object comprising a null pointer ("checked pointers, a simple wrapper for C++ pointers that prevents pointer arithmetic and other common sources of pointer errors, and detects all ...errors...The syntax of checked pointers is highly faithful to raw C++ pointers, but provides run-time error detection and debugging information," abstract; see also Class Pointer definition in Pointer_2.h of the technical addendum of this article) and

performing error processing in response to said improper native pointer object ("keep track of where each pointer is pointing, and which memory locations are currently allocated. We check at run-time that a pointer points t a legal address before deleting or dereferencing it. If the check fails, execution halts and the error is reported immediately," pg 355, left column, paragraph 4) as claimed.

12. Claims 24 and 29 are rejected under 35 U.S.C. 103(a) as being unpatentable over Pike et al. ("Checkmate: Cornering C++ Dynamic Memory Errors With Checked Pointers," 3/2000) hereinafter referred to as "Pike."

Per claim 24:

The rejection of claim 21 is incorporated, and further, Pike does not explicitly teach calling a function, which returns as its value an object of said safe pointer type. Official Notice is taken that calling a function with returning statement, within another function is well known in the art of software development, at the time applicant's invention was made. This programming technique is used to make the program easier to understand, modify, avoid redundant coding, reuse, test and debug by breaking a program's task into subtasks, solving these subtasks by sub-algorithms and then calling the subtasks. It would have been obvious for one skilled in the art of computer software development to modify Pike's disclosed T& operator* () function which returns as its value an object of the Pointer class to reuse the T* operator->() by simply calling this function and dereferencing the returned pointer to an object, as this function has a same task of "detect[ing] and report[ing] common bugs such as illegal dereferences (pg 352, left

column)" with that in T* operator->(). The modification would be obvious because one skilled in the art would be motivated to reuse the same code portions without repeating the same in another function.

Pike further discloses: object operable to encapsulate a misaligned pointer as an improper native pointer object; performing said standard pointer operation upon said object ("checked pointers, a simple wrapper for C++ pointers that prevents pointer arithmetic and other common sources of pointer errors, and detects all ...errors... The syntax of checked pointers is highly faithful to raw C++ pointers, but provides run-time error detection and debugging information," abstract; see also Class Pointer definition in Pointer_2.h of the technical addendum of this article) and performing error processing in response to said improper native pointer object ("keep track of where each pointer is pointing, and which memory locations are currently allocated. We check at run-time that a pointer points t a legal address before deleting or dereferencing it. If the check fails, execution halts and the error is reported immediately," pg 355, left column, paragraph 4) as claimed.

Per claim 29, it is the program product version of claim 24, respectively, and is rejected for the same reasons set forth in connection with the rejection of claim 24 above.

Response to Arguments

13. Applicant's arguments with respect to claims 1, 6-8, 10, 11, and 17 have been considered but are moot in view of the new ground(s) of rejection.

14. The examiner provides a reference, Savitch ("Problem solving with C++," 1995) that teaches the fundamental top-down design programming technique, to support the rejections of claims 6 and 8 based on Official Notice, in response to the applicant's request.

Conclusion

15. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

16. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Insun Kang whose telephone number is 571-272-3724. The examiner can normally be reached on M-F 9:30-6.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on 571-272-3719. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

I. Kang
Examiner
2/7/2005

KAKALI CHAKI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100